**Modalità di utilizzo del codice**

**Query**

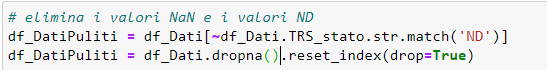
Il codice è stato scritto utilizzando i Notebook Jupyter, integrati con librerie pandas, numpy, matplotlib e excel su linguaggio di programmazione python.

Le query sono state implementate usando Python con l’ausilio della libreria pandas.

Inizialmente è stato creato un dataframe **df\_DatiSensori** direttamente dal file excel ‘**Dati\_gruppo1.xlsx**’ e poi è stata fatto un dataframe copia **df\_Dati** per lavorarci.



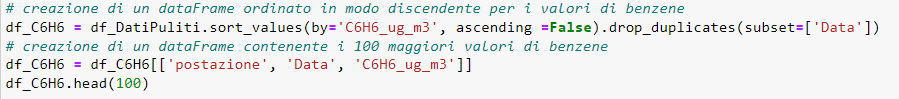
Successivamente effettuiamo una pulizia del dataset appena importato. Innanzitutto andiamo a rinominare le colonne ed effettuiamo operazioni per il riconoscimento di eventuali valori 0 consecutivi che potrebbero significare fallimento dei sensori nell’invio dei dati, dei valori nulli e degli errori riportati con ‘**ND**’ nel file excel. Infine andiamo a effettuare un reset degli indici del dataframe per non avere salti tra un indice ed un altro.



**Le 100 registrazioni con il maggior livello di benzene**

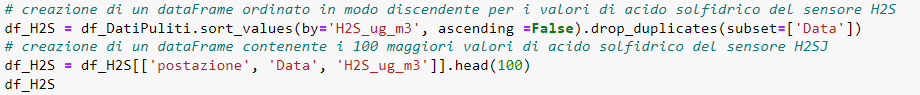
Nella prima query abbiamo creato un dataframe ordinato in modo discendete per i valori di benzene, andando ad eliminare qualsiasi duplicato riguardo alla data.

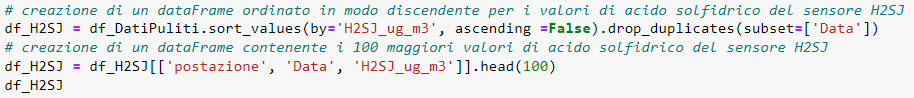
Il dataframe finale conterrà la postazione, la data e il valore del benzene, mostrando i primi cento.



**Le 100 registrazioni con il maggior livello di acido solfidrico per i sensori H2S e H2SJ**

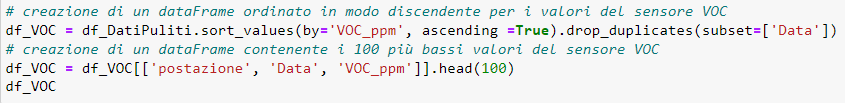
Nella seconda query creiamo due dataframe, **df\_H2S** e **df\_H2SJ**, per i due sensori ed effettuiamo le stesse operazioni effettuate per la prima query.

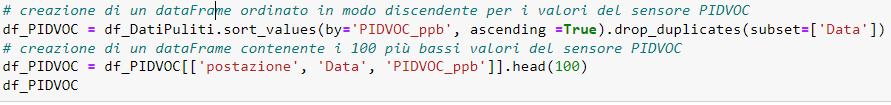




**Le 100 registrazioni con i più bassi livelli di VOC per i sensori VOC e PIDVOC**

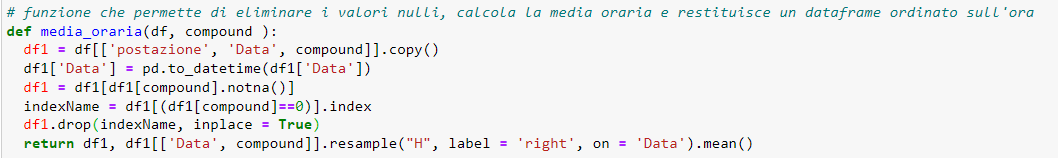
Nella terza query creiamo due dataframe, **df\_VOC** e **df\_PIDVOC**, per i due sensori ed effettuiamo le stesse operazioni effettuate per la prima query, con la differenza che in questo caso andremo ad ordinare i valori in maniera ascendente, cosi da prendere i 100 valori più bassi per i sensori.





**Le 50 ore con il più alto/basso livello medio di benzene**

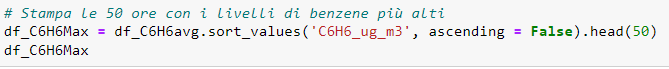
Per questa query è stata creata una funzione che permette di eliminare i valori nulli, calcolare la media oraria e restituire un dataframe ordinato sull’orae che contiene solo le colonne postazione, data e il composto interessato.

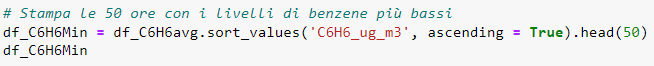


Dopodiché si è pensato di creare un dataframe richiamando la funzione **media\_oraria**, andando ad aggiungere come parametri della funzione il dataframe **df\_DatiPuliti** e il composto interessato, ovvero il benzene.



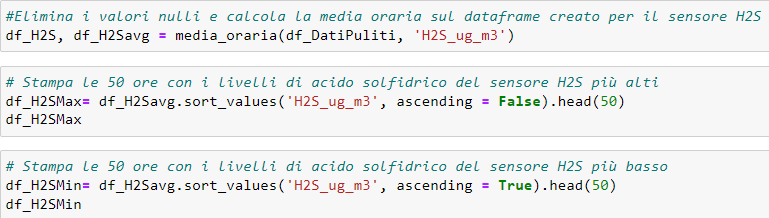
Infine è stata calcolata la più alta e la più bassa media oraria di benzene, andando poi a mostrare solo i primi 50 valori, modificando solo il parametro **ascending** della funzione **sort\_values**.

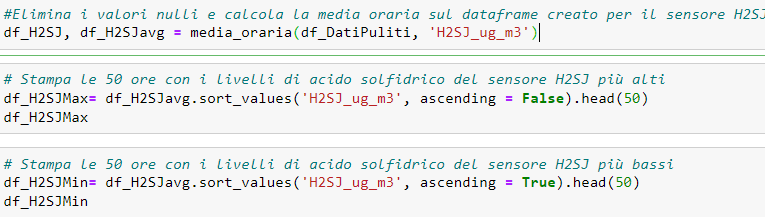




**Le 50 ore con il più alto/basso livello medio di acido solfidrico secondo i sensori H2S e H2SJ**

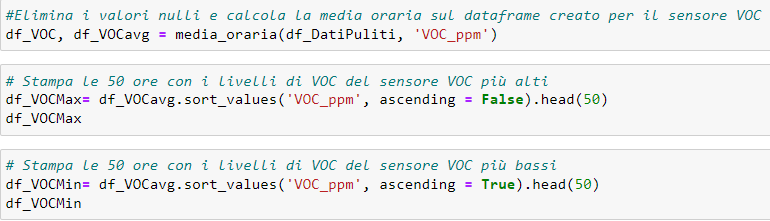
Sono state effettuate le stesse operazioni della query precedente, duplicate per entrambe i sensori **H2S** e **H2SJ**.

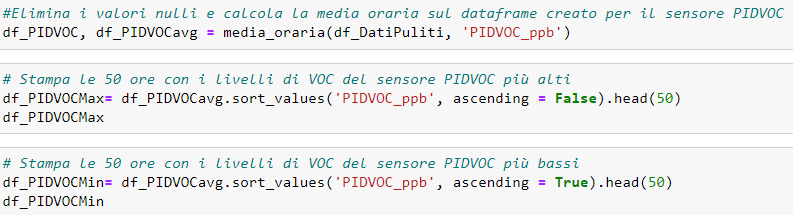




**Le 50 ore con il più alto/basso livello medio di VOC secondo i sensori VOC e PIDVOC**

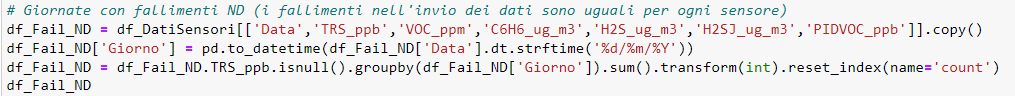
Sono state effettuate le stesse operazioni della query precedente, duplicate per entrambe i sensori **VOC** e **PIDVOC**.





**Le 3 giornate con il maggior numero di fallimenti nell’invio dei dati**

Per questa query è stato creato un dataframe df\_Fail\_ND che ha come colonne la Data e i valori numerici dei sensori. Creiamo una nuova colonna settando il tipo data a ‘giorno/mese/anno’ ed effettuiamo il conteggio dei valori NaN sulla colonna TRS\_ppb (il conteggio risulta identico su qualunque altra colonna poiché il fallimento di una stazione coincide con il fallimento di invio dei dati di ogni sensore) effettuando il groupby sul ‘Giorno’, sommando e infine resettando gli indici, creando così una nuova colonna ‘count’.



Infine è stato effettuato un ordinamento decrescente sulla colonna ‘count’ e presi in considerazione solo le prime 3 righe.



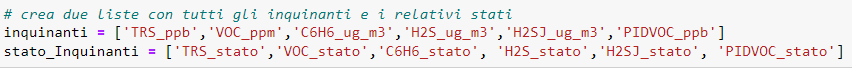
**Le 3 giornate con il minor numero di fallimenti nell’invio dei dati**

Per questa query sono state effettuate le stesse operazioni di quella precedente andando a modificare accuratamente i parametri interessati.

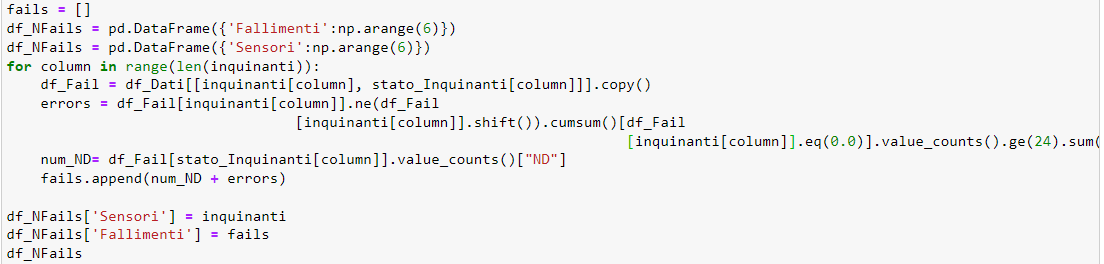


**Il numero medio di fallimenti nell’invio per sensore**

Per questa query sono state create due liste, **inquinanti** e **stato\_inquinanti**, da usare nella funzione successiva.



È stato sviluppato un metodo che permette di contare la somma dei fallimenti per ogni sensore andando poi ad inserirli in un nuovo dataframe **df\_NFails**. Innanzitutto creo una lista di fallimenti, **fails**, e poi inizializzo il nuovo dataframe **df\_NFails** a 2 colonne di nome **Sensori** e **Fallimenti**. Vado ad effettuare un ciclo sulla lunghezza di una delle due liste (quale è indifferente, avendo entrambe la stessa lunghezza). Nel ciclo creo un dataframe temporaneo contenente solo le colonne **inquinante** e **stato\_inquinante**. La variabile **errors** restituisce il numero di fallimenti dovuti alla presenza di 0 consecutivi (come fatto per la pulizia del dataset, abbiamo scelto un valore di 24 zeri consecutivi che equivalgono a 2 ore di invii di dati considerati falliti), mentre **num\_ND** restituisce il numero di valori **ND** presenti. Infine questi valori vengono sommati e aggiunti alla lista **fails**. Fuori dal ciclo for viene popolato il dataframe **df\_NFails** con la colonna sensori uguagliata alla lista inquinanti e la colonna fallimenti con **fails**.



Viene aggiunta una colonna ‘**Dati TOT**’ che comprende la lunghezza dei dati per ogni sensore ed infine viene creata un’ultima colonna chiamata ‘**Media**’ contente la media tra la colonna ‘**Fallimenti**’ e ‘**Dati TOT**’.



**Il sensore con il numero massimo di fallimenti**

Per questa query è stato utilizzato lo stesso dataframe creato nella query precedente ed è stata fatta una chiamata **max()** al dataframe **df\_NFails** sulla colonna ‘**Fallimenti**’.



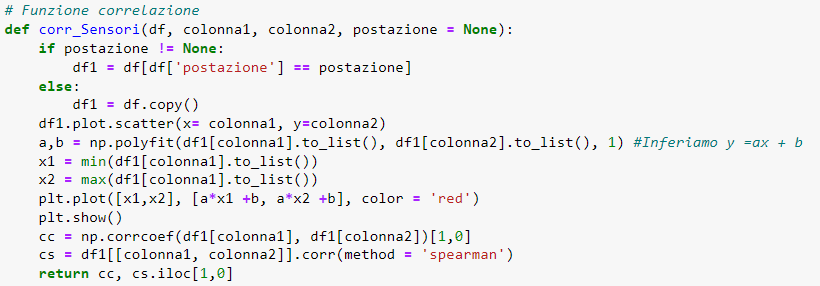
**Il sensore con il numero minimo di fallimenti**

Per questa query è stato utilizzato lo stesso dataframe creato nella query precedente ed è stata fatta una chiamata **min()** al dataframe **df\_NFails** sulla colonna ‘**Fallimenti**’



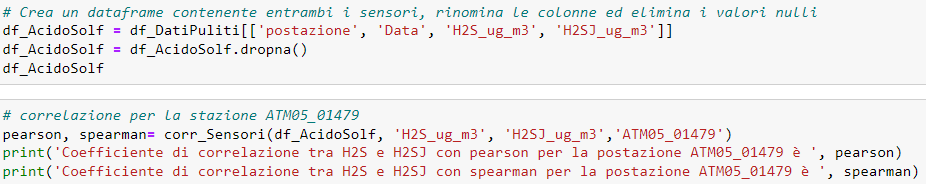
**Correlazioni**

Inizialmente è stata creata una funzione correlazione **corr\_Sensori** che mi permette di ricevere in output due valori, ovvero la correlazione di **Pearson** e la correlazione di **Spearman**. Questa funzione prende in input il dataframe su cui effettuare le correlazioni, i due composti come stringa che si intendono mettere in correlazione e la postazione, anch’essa stringa, che è settata come **None** (dato che questa funzione è stata adattata per essere utilizzata per due tipi di correlazioni diverse). Inizialmente verifica che il valore di **postazione** è diverso da **None** cosi da poter creare un dataframe **df1** prendendo solo la colonna postazione interessata, altrimenti fa una copia dell’intero dataframe che è stato passato. Viene effettuata la regressione lineare e graficati i valori e calcoliamo i coefficienti di **Pearson** e di **Spearman**.



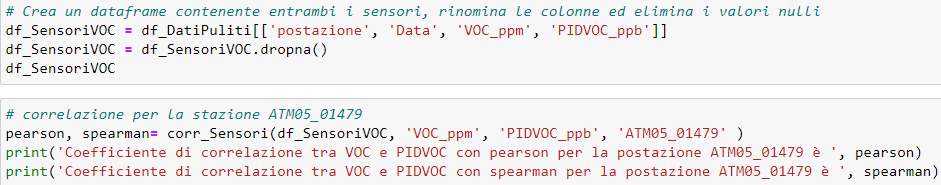
**Correlazione tra H2S e H2SJ in una data stazione**

Nella prima correlazione creiamo un dataframe df\_AcidoSolf prendendo dal dataframe df\_DatiPuliti le colonne di **postazione**, **Data**, **H2S\_ug\_m3** e **H2SJ\_ug\_m3**, effettuiamo l’eliminazione dei valori nulli attraverso la funzione **dropna()** e infine richiamiamo la funzione **corr\_Sensori** passando le come variabili il dataframe **df\_AcidoSolf**, **H2S\_ug\_m3**, **H2SJ\_ug\_m3** e la **postazione** (viene effettuata su tutte e 4). I valori ottenuti vengono associati alle variabili **pearson** e **spearman** e poi stampate a video.



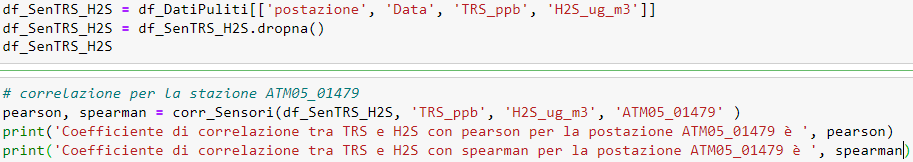
**Correlazione tra VOC e PIDVOC in una data stazione**

Per questa correlazione sono state effettuate le stesse operazioni della correlazione precedente andando ad effettuare le opportune modifiche sui composti interessati, **VOC** e **PIDVOC**.



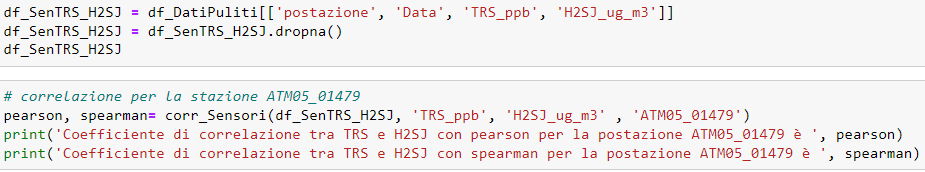
**Correlazione tra TRS e H2S in una data stazione**

Per questa correlazione sono state effettuate le stesse operazioni della correlazione precedente andando ad effettuare le opportune modifiche sui composti interessati, **TRS** e **H2S**.

****

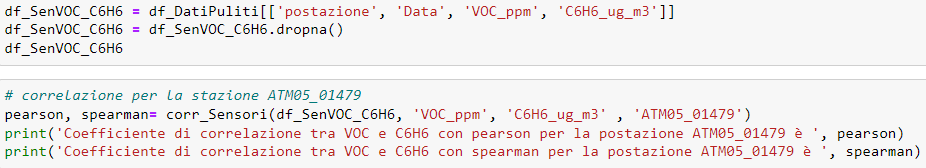
**Correlazione tra TRS e H2SJ in una data stazione**

Per questa correlazione sono state effettuate le stesse operazioni della correlazione precedente andando ad effettuare le opportune modifiche sui composti interessati, **TRS** e **H2SJ**.

****

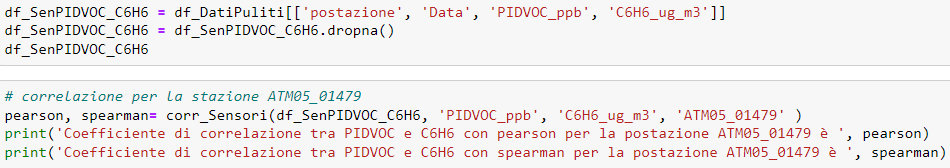
**Correlazione tra VOC e C6H6 in una data stazione**

Per questa correlazione sono state effettuate le stesse operazioni della correlazione precedente andando ad effettuare le opportune modifiche sui composti interessati, **VOC** e **C6H6**.

****

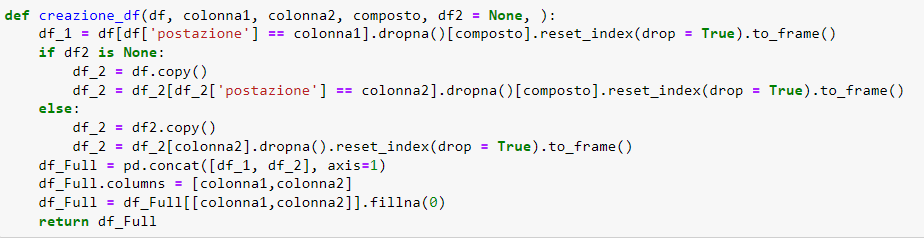
**Correlazione tra PIDVOC e C6H6 in una data stazione**

Per questa correlazione sono state effettuate le stesse operazioni della correlazione precedente andando ad effettuare le opportune modifiche sui composti interessati, **PIDVOC** e **C6H6**.

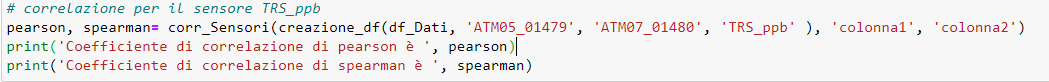


**Correlazione in stazioni diverse per lo stesso sensore**

Per questa correlazione è stata sviluppata una funzione che permette di creare e restituire un dataframe costituito dalle colonne necessarie. Le colonne sono 2 e sono chiamate ‘**colonna1**’ e ‘**colonna2**’ a causa della riusabilità del codice della funzione (utilizzato anche per correlazioni che hanno diverse finalità). La funzione **creazione\_df** prende in input un dataframe **df**, le due colonne, il composto e poi presenta un parametro di default, **df2 = None**,per l’elaborazione delle correlazioni successive. Viene inizializzato il primo dataframe temporaneo **df\_1** prendendo le righe con postazione uguale al valore di **colonna1**, eliminati i valori e la colonna corrispondente al composto. Infine vengono resettati gli indici e reso un dataframe. Viene effettuato il controllo su **df2**, per verificare se è stato passato come parametro. Infine viene creato il dataframe **df\_Full** concatenando i due dataframe, rinominate le colonne e sostituiti i valori **NaN** con 0.



Per effettuare la correlazione viene richiamata la funzione **corr\_Sensori** e passati, come parametri, la funzione **creazione\_df** con i suoi parametri e la **colonna1** e la **colonna2** come stringa. Infine vengono visualizzate le correlazioni di **Pearson** e **Spearman**. Questa operazione viene effettuata per tutte le combinazioni di sensori e stazioni.

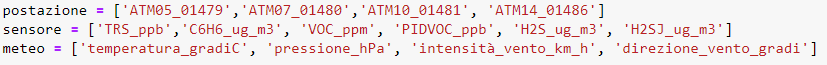


**Correlazione con la temperatura per ogni stazione e sensore**

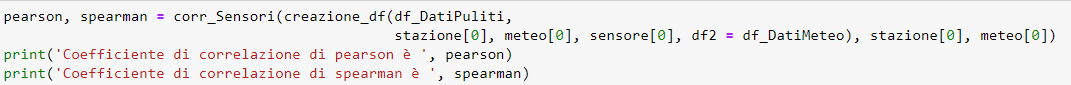
Innanzitutto carichiamo il dataset dei dati meteo relativi alla zona di analisi.

****

Creiamo tre liste, una con tutte le postazioni, una con tutti i sensori e una con tutte le informazioni meteo necessarie.

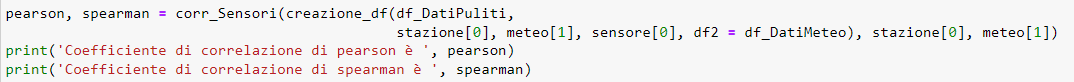


A questo punto viene richiamata la funzione **corr\_Sensori** con i parametri relativi al dataframe, richiamando la funzione **creazione\_df**, la **colonna** **1** e la **colonna2**. Per la creazione del dataframe sono stati utilizzati il **df\_DatiPuliti** per **df**, **postazione[ ]** per indicare la stazione in **df\_DatiPuliti, meteo[0]** per indicare le informazioni meteo da correlare, ovvero la temperatura, **sensore[ ]** per indicare il composto da analizzare e infine il dataframe **df\_DatiMeteo**. Infine sono state mostrate a video le correlazioni di **Pearson** e **Spearman**.



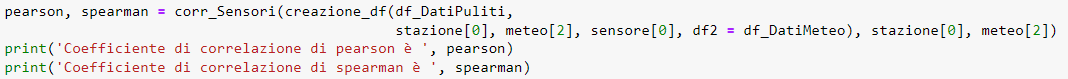
**Correlazione con la direzione del vento per ogni stazione e sensore**

Per questa correlazione sono state effettuate le stesse operazione per la correlazione precedente andando a modificare opportunamente i valori necessari, ovvero **meteo [1]** per la direzione del vento.



**Correlazione con la pressione atmosferica per ogni stazione e sensore**

Per questa correlazione sono state effettuate le stesse operazione per la correlazione precedente andando a modificare opportunamente i valori necessari, ovvero **meteo [2]** per la pressione atmosferica.



**Correlazione con l’intensità del vento per ogni stazione e sensore**

Per questa correlazione sono state effettuate le stesse operazione per la correlazione precedente andando a modificare opportunamente i valori necessari, ovvero **meteo [3]** per l’intensità del vento.

